DJILLALI LIABES UNIVERSITY OF SIDI BEL ABBES
FACULTY OF EXACT SCIENCES
DEPARTMENT OF COMPUTER SCIENCES

Module : Réseaux et Systèmes Répartis
1ST YEAR OF MASTER'S DEGEREE IN
NETWORKS,INFORMATION SYSTEMS & SECURITY(RSSI)
2021/2022

Application client-serveur avec les sockets UDP java pour la recherche d'un mot dans texte

---

Students:
HADJAZI M.Hisham
AMOUR Wassim Malik
Group: 01/RSSI

Instructors:
Dr. MEHADJI Djamil

A paper submitted in fulfilment of the requirements for the
TP-02

March 20, 2022

Contents

Chapter 1

Application client-serveur avec les sockets TCP java

1.1 Introduction

UDP is a communication protocol that transmits independent packets over the network with no guarantee of arrival and no guarantee of the order of delivery.

Most communication over the internet takes place over the Transmission Control Protocol (TCP), however, UDP has its place which we will be exploring in the next section.

1.1.1 Why Use UDP?

UDP is quite different from the more common TCP. But before considering the surface level disadvantages of UDP, it's important to understand that the lack of overhead can make it significantly faster than TCP.

Apart from speed, we also need to remember that some kinds of communication do not require the reliability of TCP but value low latency instead. The video is a good example of an application that might benefit from running over UDP instead of TCP.

1.1.2 Building UDP Applications

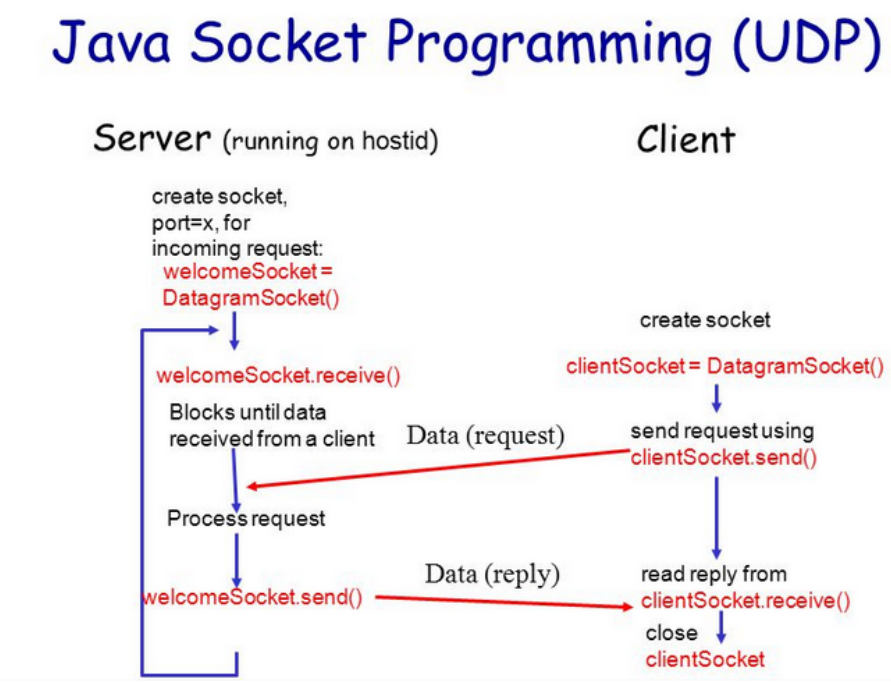

FIGURE 1.1: UDP Socket in JAVA.

Building UDP applications is very similar to building a TCP system; the only difference is that we don't establish a point to point connection between a client and a server.

The setup is very straightforward too. Java ships with built-in networking support for UDP – which is part of the java.net package. Therefore to perform networking operations over UDP, we only need to import the classes from the java.net package: java.net.DatagramSocket and java.net.DatagramPacket.

In the following sections, we will learn how to design applications that communicate over UDP; we'll use the popular echo protocol for this application.

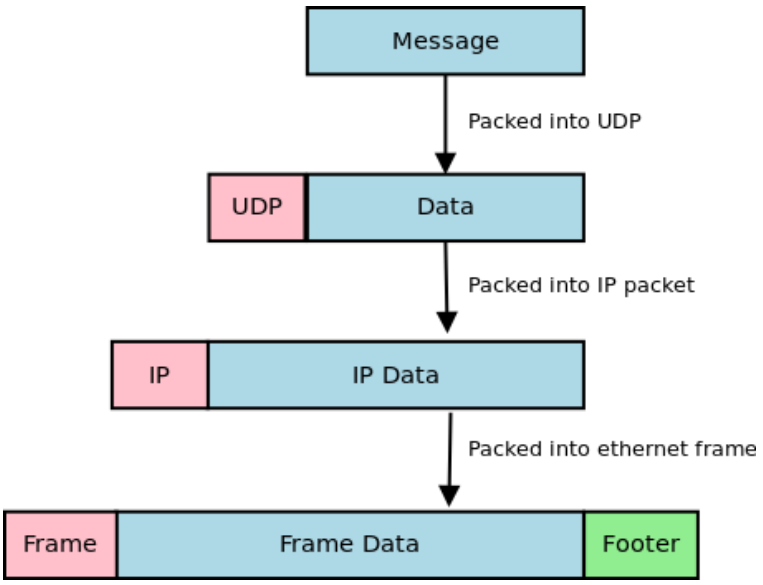

FIGURE 1.2: UDP Socket Communication Flow Diagram.

First, we will build an echo server that sends back any message sent to it, then an echo client that just sends any arbitrary message to the server and finally, we will test the application to ensure everything is working fine.

1.2 Implementation in Client

1.2.1 Imports used

```
1 import java.net.DatagramPacket;
2 import java.net.DatagramSocket;
3 import java.net.InetAddress;
4 import java.net.SocketException;
```

1.2.2 Client class

We have our global DatagramSocket and address of the server. We instantiate these inside the constructor.

We have a separate method which sends messages to the server and returns the response.

We first convert the string message into a byte array, then create a DatagramPacket for sending messages.

Next – we send the message. We immediately convert the DatagramPacket into a receiving one.

When the echo arrives, we convert the bytes to a string and return the string.

```
5  static class EchoClient {
6
7          private DatagramSocket socket;
8          private InetAddress address;
9          private int port;
10
11         private byte[] buf = new byte[65535];
12
13         public EchoClient(InetAddress adr, int por) throws
               SocketException, UnknownHostException {
14             socket = new DatagramSocket();
15             address = adr;
16             port = por;
17         }
18
19         public String sendEcho(String msg) throws Exception {
20             buf = msg.getBytes();
21             DatagramPacket packet = new DatagramPacket(buf, buf.
                   length, address, port);
22             socket.send(packet);
23             buf = new byte[65535];
24             packet = new DatagramPacket(buf, buf.length);
25             socket.receive(packet);
```

```
26              String received = new String(packet.getData(), 0,
                    packet.getLength());
27              return received;
28          }
29
30          public void close() {
31              socket.close();
32          }
33      }
```

1.2.3 Initialisation of Client

```
34  EchoClient client = new EchoClient(add, SERVER_PORT);
```

1.2.4 GUI of Client

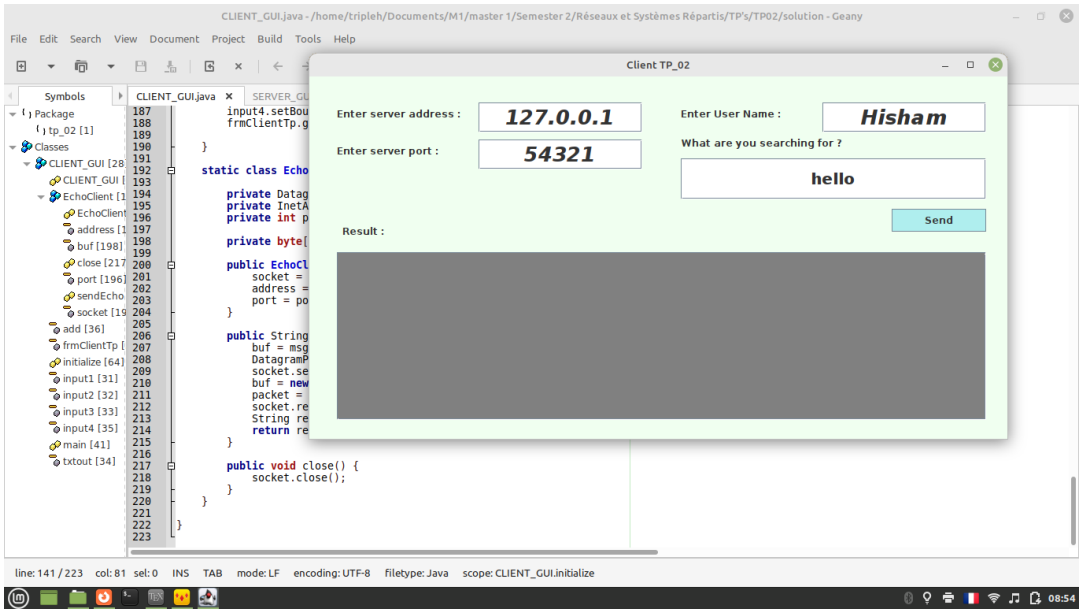

FIGURE 1.3: Client GUI.

1.3 Implementation in Server

In UDP communication, a single message is encapsulated in a DatagramPacket which is sent through a DatagramSocket.

We create a global DatagramSocket which we will use throughout to send packets, a byte array to wrap our messages, and a status variable called running.

For simplicity, the server is extending Thread, so we can implement everything inside the run method.

Inside run, we create a while loop that just runs until running is changed to false by some error or a termination message from the client.

At the top of the loop, we instantiate a DatagramPacket to receive incoming messages.

Next, we call the receive method on the socket. This method blocks until a message arrives and it stores the message inside the byte array of the DatagramPacket passed to it.

After receiving the message, we retrieve the address and port of the client, since we are going to send the response back.

Next, we create a DatagramPacket for sending a message to the client. Notice the difference in signature with the receiving packet. This one also requires address and port of the client we are sending the message to.

1.3.1 Imports used

```
35 import java.net.DatagramPacket;
36 import java.net.DatagramSocket;
37 import java.net.InetAddress;
38 import java.net.SocketException;
```

1.3.2 Server class

```
39 static class EchoServer extends Thread {
40
41         private boolean running;
42         private byte[] buf = new byte[65535];
43         private String str;
44         private int ctr = 0;
45         private String[][] Sresults = new String[100][2];
46         private String msg = new String("");
47
48         public EchoServer() throws SocketException {
49             int PORT = Integer.parseInt(input.getText());
50             socket = new DatagramSocket(PORT);
51         }
52
53         public void run() {
54             running = true;
55
56             txtout.setFont(new Font("Dialog", Font.BOLD, 9));
57             txtout.setForeground(new Color(0, 255, 0));
58             txtout.append("SERVER : Port is open\nSERVER :
                listening....");
59
60             while (running) {
61                 DatagramPacket packet = new DatagramPacket(buf, buf
                    .length);
62                 try {
63                     socket.receive(packet);
64
```

```
65                  str = new String(packet.getData(), 0, packet.
                        getLength());
66                  System.out.println("\nRecieved " + str);
67
68              } catch (IOException e) {
69                  // TODO Auto-generated catch block
70                  e.printStackTrace();
71              }
72
73              // Do stuff to the received str plus prepare
                    statements
74
75              // split username and searched word
76
77              String[] results = str.split("\\s*,\\s*");
78
79              System.out.println("\nwe got " + results[0] + "
                    searche for  " + results[1]);
80
81              txtout.setFont(new Font("Dialog", Font.BOLD, 9));
82              txtout.setForeground(new Color(0, 255, 0));
83              txtout.append("\nRecieved " + str + "\nProcessing
                    ....");
84
85              // find number of occurances of a word
86
87              String[] words = text.split(" ");
88              String word = results[1];
89              int occ = 0;
90              for (int i = 0; i < words.length; i++) {
91                  if (words[i].equals(word)) {
92                      occ++;
93                  }
94              }
95              System.out.println("\nnumber of occurences = " +
                    occ);
96
97              msg = ("Number of occurences for the word : '" +
                    word + "' is = " + occ + " times.\n\n");
98
99              // add user and the searched for word
100
101             Sresults[ctr][0] = results[0];
102             Sresults[ctr][1] = results[1];
103
104             System.out.println(Sresults[ctr][0] + " searched
                    for " + Sresults[ctr][1]);
105
106             // query db for all users who searched for the same
                     word
107
```

```
108                 for (int i = 0; i <= ctr; i++) {
109
110                     if (Sresults[i][1].equals(results[1])) {
111
112                         msg = msg + ("Client " + Sresults[i][0] + "
                                , Searched For : " + Sresults[i][1] + "
                               \n");
113                     }
114                 }
115
116                 System.out.println(msg);
117                 buf = msg.getBytes();
118                 InetAddress address = packet.getAddress();
119                 int port = packet.getPort();
120                 packet = new DatagramPacket(buf, buf.length,
                        address, port);
121
122                 ctr++;
123
124                 try {
125                     socket.send(packet);
126                 } catch (IOException e) {
127                     // TODO Auto-generated catch block
128                     e.printStackTrace();
129                 }
130             }
131             socket.close();
132         }
133     }
```

1.3.3 Find Number of Occurrences

```
134 // find number of occurances of a word
135
136                 String[] words = text.split(" ");
137                 String word = results[1];
138                 int occ = 0;
139                 for (int i = 0; i < words.length; i++) {
140                     if (words[i].equals(word)) {
141                         occ++;
142                     }
143                 }
```

1.3.4 Initialisation of Server

```
144 new EchoServer().start();
```

1.3.5 GUI of Server

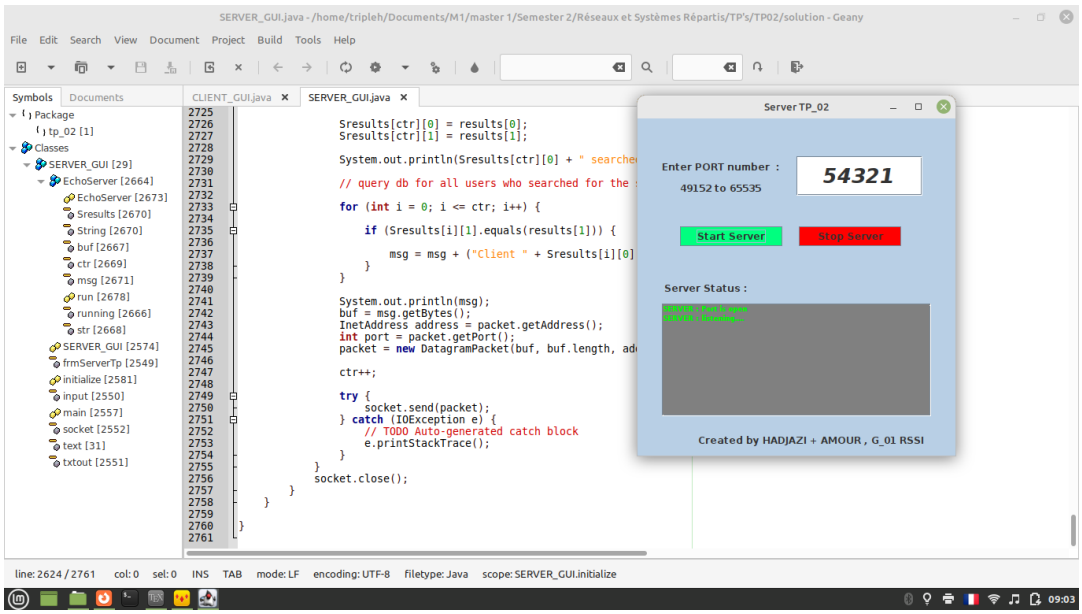

FIGURE 1.4: Server GUI.

1.4 Execution of client and server

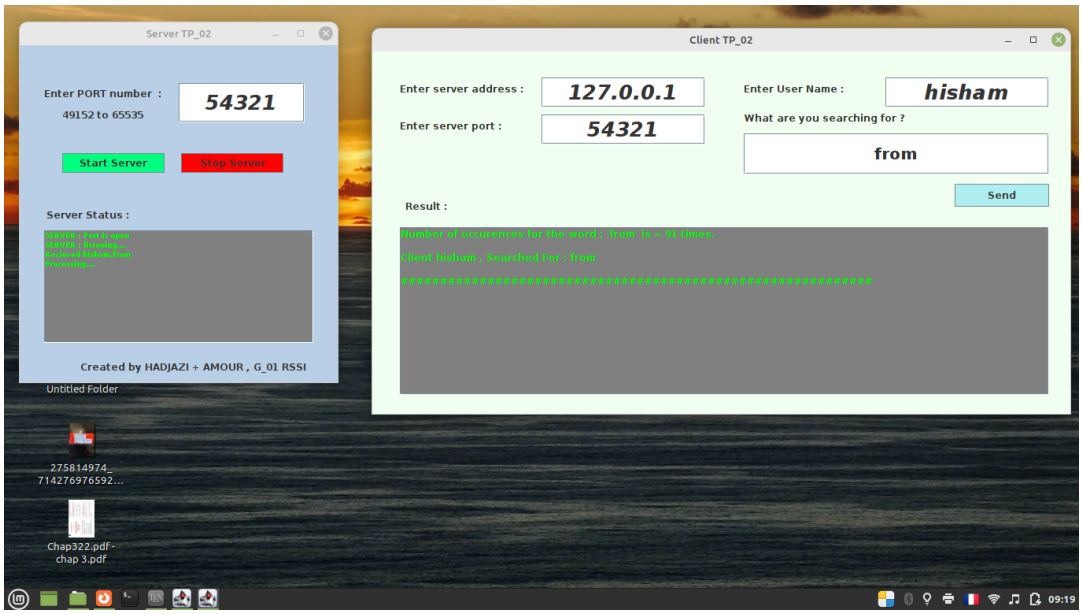

FIGURE 1.5: Execution 1.

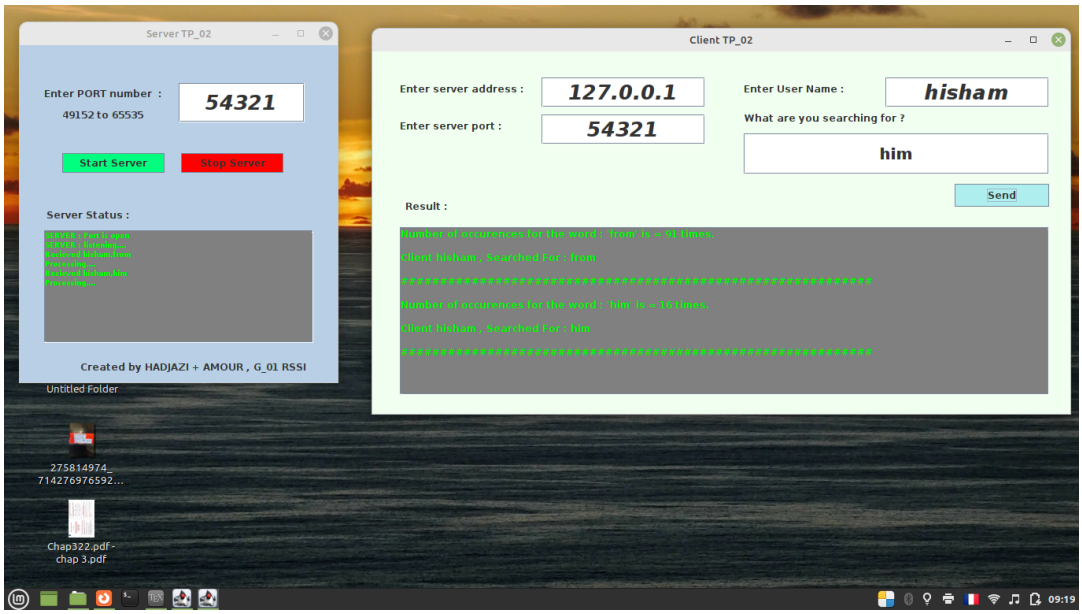

FIGURE 1.6: Execution 2.

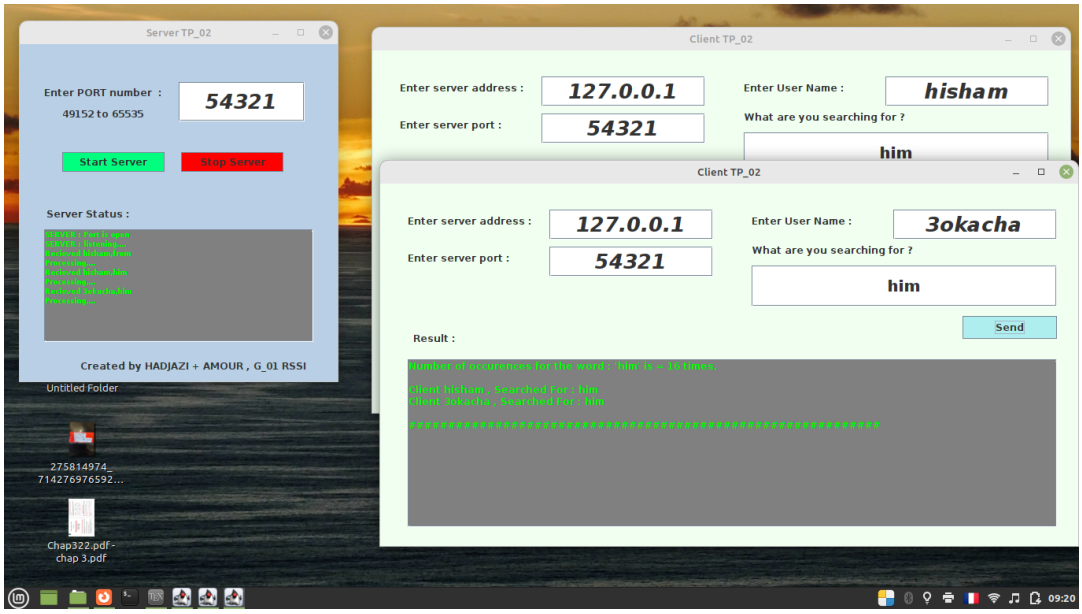

FIGURE 1.7: Execution 3.

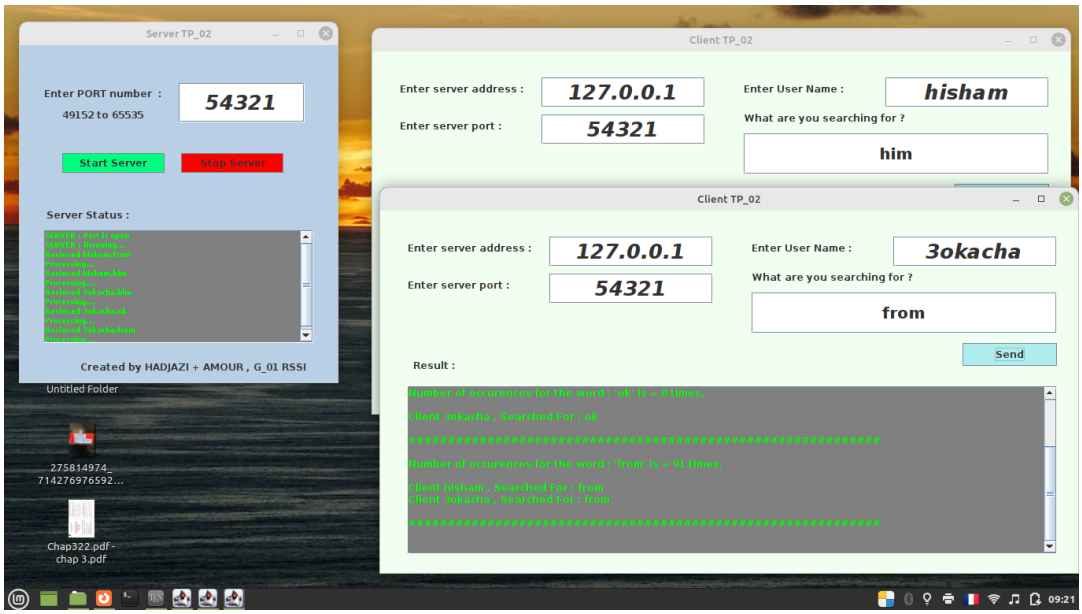

FIGURE 1.8: Execution 4.

Appendix A

Appendix A

A.1 Java Code for SERVER-GUI.java

```
145 package tp_02;
146
147 //TP2 Reseaux et Systemes Repartis 2021-2022
148
149 //Nom:HADJAZI
150 //Prenom: Mohammed Hisham
151 //Specialite:   RSSI      Groupe: 01
152
153 import java.awt.EventQueue;
154 import javax.swing.JFrame;
155 import javax.swing.JTextField;
156 import javax.swing.JLabel;
157 import javax.swing.SwingConstants;
158 import javax.swing.SwingWorker;
159 import java.awt.Font;
160 import java.awt.Color;
161 import javax.swing.JButton;
162 import javax.swing.UIManager;
163 import java.awt.event.ActionListener;
164 import java.io.IOException;
165 import java.net.DatagramPacket;
166 import java.net.DatagramSocket;
167 import java.net.InetAddress;
168 import java.net.SocketException;
169 import java.awt.event.ActionEvent;
170 import javax.swing.JTextArea;
171 import javax.swing.JScrollPane;
172
173 public class SERVER_GUI {
174
175     private static String text = " THE LETTERS OF A PORTUGUESE NUN"
        ;
176     private JFrame frmServerTp;
177     private static JTextField input;
178     static JTextArea txtout;
179     static DatagramSocket socket;
180
181     /**
182      * Launch the application.
183      */
184     public static void main(String[] args) {
185
```

```
186         EventQueue.invokeLater(new Runnable() {
187             public void run() {
188                 try {
189                     SERVER_GUI window = new SERVER_GUI();
190                     window.frmServerTp.setVisible(true);
191                 } catch (Exception e) {
192                     e.printStackTrace();
193                 }
194             }
195         });
196     }
197 
198     /**
199      * Create the application.
200      */
201     public SERVER_GUI() {
202         initialize();
203     }
204 
205     /**
206      * Initialize the contents of the frame.
207      */
208     private void initialize() {
209         frmServerTp = new JFrame();
210         frmServerTp.setBackground(new Color(64, 224, 208));
211         frmServerTp.getContentPane().setBackground(UIManager.
                getColor("activeCaption"));
212         frmServerTp.setTitle("Server TP_02");
213         frmServerTp.setBounds(100, 100, 406, 459);
214         frmServerTp.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
215         frmServerTp.getContentPane().setLayout(null);
216 
217         input = new JTextField();
218         input.setFont(new Font("Dialog", Font.BOLD | Font.ITALIC,
                26));
219         input.setHorizontalAlignment(SwingConstants.CENTER);
220         input.setBounds(202, 48, 160, 50);
221         frmServerTp.getContentPane().add(input);
222         input.setColumns(10);
223 
224         JLabel lblEnterPortNumber = new JLabel("Enter PORT number
                :");
225         lblEnterPortNumber.setBounds(31, 41, 178, 40);
226         frmServerTp.getContentPane().add(lblEnterPortNumber);
227 
228         JLabel lblTo = new JLabel("49152 to 65535");
229         lblTo.setBounds(54, 68, 130, 40);
230         frmServerTp.getContentPane().add(lblTo);
231 
232         JLabel lblServerStatus = new JLabel("Server Status :");
233         lblServerStatus.setBounds(34, 208, 135, 15);
234         frmServerTp.getContentPane().add(lblServerStatus);
235 
236         JButton btnStart = new JButton("Start Server");
237         btnStart.addActionListener(new ActionListener() {
238             public void actionPerformed(ActionEvent arg0) {
239 
```

```
240                 new SwingWorker() {
241
242                     @Override
243                     protected Object doInBackground() throws
                            Exception {
244
245                         new EchoServer().start();
246                         return null;
247
248                     }
249
250                 }.execute();
251
252             }
253
254         });
255         btnStart.setBackground(new Color(0, 255, 127));
256         btnStart.setBounds(54, 137, 130, 25);
257         frmServerTp.getContentPane().add(btnStart);
258
259         JButton btnStop = new JButton("Stop Server");
260         btnStop.addActionListener(new ActionListener() {
261             public void actionPerformed(ActionEvent arg0) {
262
263                 // Close server socket
264
265                 txtout.setFont(new Font("Dialog", Font.BOLD, 9));
266                 txtout.setForeground(new Color(0, 255, 0));
267                 txtout.append("\nSERVER : Port is Closed");
268                 socket.close();
269
270             }
271         });
272         btnStop.setBackground(new Color(255, 0, 0));
273         btnStop.setBounds(205, 137, 130, 25);
274         frmServerTp.getContentPane().add(btnStop);
275
276         JLabel lblCreatedByHadjazi = new JLabel("Created by HADJAZI
                + AMOUR , G_01 RSSI");
277         lblCreatedByHadjazi.setBounds(77, 401, 293, 15);
278         frmServerTp.getContentPane().add(lblCreatedByHadjazi);
279
280         JScrollPane scrollPane = new JScrollPane();
281         scrollPane.setBounds(31, 235, 342, 143);
282         frmServerTp.getContentPane().add(scrollPane);
283
284         txtout = new JTextArea();
285         scrollPane.setViewportView(txtout);
286         txtout.setForeground(Color.GREEN);
287         txtout.setBackground(Color.GRAY);
288
289     }
290
291     static class EchoServer extends Thread {
292
293         private boolean running;
294         private byte[] buf = new byte[65535];
```

```
295         private String str;
296         private int ctr = 0;
297         private String[][] Sresults = new String[100][2];
298         private String msg = new String("");
299 
300         public EchoServer() throws SocketException {
301             int PORT = Integer.parseInt(input.getText());
302             socket = new DatagramSocket(PORT);
303         }
304 
305         public void run() {
306             running = true;
307 
308             txtout.setFont(new Font("Dialog", Font.BOLD, 9));
309             txtout.setForeground(new Color(0, 255, 0));
310             txtout.append("SERVER : Port is open\nSERVER :
                listening....");
311 
312             while (running) {
313                 DatagramPacket packet = new DatagramPacket(buf, buf
                    .length);
314                 try {
315                     socket.receive(packet);
316 
317                     str = new String(packet.getData(), 0, packet.
                        getLength());
318                     System.out.println("\nRecieved " + str);
319 
320                 } catch (IOException e) {
321                     // TODO Auto-generated catch block
322                     e.printStackTrace();
323                 }
324 
325                 // Do stuff to the received str plus prepare
                    statements
326 
327                 // split username and searched word
328 
329                 String[] results = str.split("\\s*,\\s*");
330 
331                 System.out.println("\nwe got " + results[0] + "
                    searche for  " + results[1]);
332 
333                 txtout.setFont(new Font("Dialog", Font.BOLD, 9));
334                 txtout.setForeground(new Color(0, 255, 0));
335                 txtout.append("\nRecieved " + str + "\nProcessing
                    ....");
336 
337                 // find number of occurances of a word
338 
339                 String[] words = text.split(" ");
340                 String word = results[1];
341                 int occ = 0;
342                 for (int i = 0; i < words.length; i++) {
343                     if (words[i].equals(word)) {
344                         occ++;
345                     }
```

```
346                 }
347                 System.out.println("\nnumber of occurences = " +
                        occ);
348
349                 msg = ("Number of occurences for the word : '" +
                        word + "' is = " + occ + " times.\n\n");
350
351                 // add user and the searched for word
352
353                 Sresults[ctr][0] = results[0];
354                 Sresults[ctr][1] = results[1];
355
356                 System.out.println(Sresults[ctr][0] + " searched
                        for " + Sresults[ctr][1]);
357
358                 // query db for all users who searched for the same
                         word
359
360                 for (int i = 0; i <= ctr; i++) {
361
362                     if (Sresults[i][1].equals(results[1])) {
363
364                         msg = msg + ("Client " + Sresults[i][0] + "
                                , Searched For : " + Sresults[i][1] + "
                               \n");
365                     }
366                 }
367
368                 System.out.println(msg);
369                 buf = msg.getBytes();
370                 InetAddress address = packet.getAddress();
371                 int port = packet.getPort();
372                 packet = new DatagramPacket(buf, buf.length,
                        address, port);
373
374                 ctr++;
375
376                 try {
377                     socket.send(packet);
378                 } catch (IOException e) {
379                     // TODO Auto-generated catch block
380                     e.printStackTrace();
381                 }
382             }
383             socket.close();
384         }
385     }
386
387 }
```

A.2 Java Code for CLIENT-GUI.java

```
389 package tp_02;
390
391 //TP2 Reseaux et Systemes Repartis 2021-2022
```

```
392
393 //Nom:HADJAZI
394 //Prenom: Mohammed Hisham
395 //Specialite:   RSSI      Groupe: 01
396
397 import java.awt.EventQueue;
398 import javax.swing.JFrame;
399 import javax.swing.JTextField;
400 import javax.swing.JLabel;
401 import javax.swing.SwingConstants;
402 import javax.swing.SwingWorker;
403 import javax.swing.JButton;
404 import java.awt.Font;
405 import java.awt.Color;
406 import java.awt.event.ActionListener;
407 import java.net.DatagramPacket;
408 import java.net.DatagramSocket;
409 import java.net.InetAddress;
410 import java.net.SocketException;
411 import java.net.UnknownHostException;
412 import java.awt.event.ActionEvent;
413 import javax.swing.JTextArea;
414 import javax.swing.JScrollPane;
415
416 public class CLIENT_GUI {
417
418     private JFrame frmClientTp;
419     private JTextField input1;
420     private JTextField input2;
421     private JTextField input3;
422     static JTextArea txtout;
423     private JTextField input4;
424     private InetAddress add;
425
426     /**
427      * Launch the application.
428      */
429     public static void main(String[] args) {
430         EventQueue.invokeLater(new Runnable() {
431             public void run() {
432                 try {
433                     CLIENT_GUI window = new CLIENT_GUI();
434                     window.frmClientTp.setVisible(true);
435                 } catch (Exception e) {
436                     e.printStackTrace();
437                 }
438             }
439         });
440     }
441
442     /**
443      * Create the application.
444      */
445     public CLIENT_GUI() {
446         initialize();
447     }
448
```

```
449     /**
450      * Initialize the contents of the frame.
451      */
452     private void initialize() {
453         frmClientTp = new JFrame();
454         frmClientTp.getContentPane().setBackground(new Color(240,
                255, 240));
455         frmClientTp.setBackground(new Color(102, 205, 170));
456         frmClientTp.setTitle("Client TP_02");
457         frmClientTp.setBounds(100, 100, 889, 491);
458         frmClientTp.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
459         frmClientTp.getContentPane().setLayout(null);
460 
461         input1 = new JTextField();
462         input1.setText("127.0.0.1");
463         input1.setFont(new Font("Dialog", Font.BOLD | Font.ITALIC,
                26));
464         input1.setHorizontalAlignment(SwingConstants.CENTER);
465         input1.setBounds(215, 33, 208, 38);
466         frmClientTp.getContentPane().add(input1);
467         input1.setColumns(10);
468 
469         input2 = new JTextField();
470         input2.setFont(new Font("Dialog", Font.BOLD | Font.ITALIC,
                26));
471         input2.setHorizontalAlignment(SwingConstants.CENTER);
472         input2.setColumns(10);
473         input2.setBounds(215, 80, 208, 38);
474         frmClientTp.getContentPane().add(input2);
475 
476         JLabel lblNewLabel = new JLabel("Enter server address :");
477         lblNewLabel.setHorizontalAlignment(SwingConstants.LEFT);
478         lblNewLabel.setBounds(35, 33, 159, 29);
479         frmClientTp.getContentPane().add(lblNewLabel);
480 
481         JLabel lblEnterServerPort = new JLabel("Enter server port :
                ");
482         lblEnterServerPort.setHorizontalAlignment(SwingConstants.
                LEFT);
483         lblEnterServerPort.setBounds(35, 80, 159, 29);
484         frmClientTp.getContentPane().add(lblEnterServerPort);
485 
486         JLabel label = new JLabel("");
487         label.setBounds(35, 130, 208, 40);
488         frmClientTp.getContentPane().add(label);
489 
490         JLabel lblResult = new JLabel("Result :");
491         lblResult.setHorizontalAlignment(SwingConstants.CENTER);
492         lblResult.setBounds(12, 182, 114, 29);
493         frmClientTp.getContentPane().add(lblResult);
494 
495         input3 = new JTextField();
496         input3.setFont(new Font("Dialog", Font.BOLD, 20));
497         input3.setHorizontalAlignment(SwingConstants.CENTER);
498         input3.setBounds(472, 104, 388, 52);
499         frmClientTp.getContentPane().add(input3);
500         input3.setColumns(10);
```

```
501
502         JLabel lblEnterNumbers = new JLabel("What are you searching
                for ?");
503         lblEnterNumbers.setBounds(472, 77, 388, 15);
504         frmClientTp.getContentPane().add(lblEnterNumbers);
505
506         JButton btnSend = new JButton("Send");
507         btnSend.addActionListener(new ActionListener() {
508             public void actionPerformed(ActionEvent arg0) {
509
510                 int SERVER_PORT = Integer.parseInt(input2.getText()
                    );
511                 try {
512                     add = InetAddress.getByName(input1.getText());
513                 } catch (UnknownHostException e1) {
514
515                     e1.printStackTrace();
516                     System.out.println("ERROR : " + e1);
517
518                 }
519
520                 new SwingWorker() {
521
522                     @Override
523                     protected Object doInBackground() throws
                        Exception {
524
525                         // Sending and Receiving solution from
                            server
526
527                         try {
528
529                             EchoClient client = new EchoClient(add,
                                SERVER_PORT);
530
531                             String UserName = input4.getText();
532                             String Search = input3.getText();
533
534                             String msg = client.sendEcho(UserName +
                                "," + Search);
535
536                             System.out.println(msg);
537
538                             txtout.setFont(new Font("Dialog", Font.
                                BOLD, 9));
539                             txtout.setForeground(new Color(0, 255,
                                0));
540                             txtout.append(msg);
541
542                         } catch (Exception e) {
543                             e.printStackTrace();
544                         }
545
546                         return null;
547                     }
548
549                 }.execute();
```

```
550
551             }
552         });
553         btnSend.setBackground(new Color(175, 238, 238));
554         btnSend.setBounds(740, 168, 120, 29);
555         frmClientTp.getContentPane().add(btnSend);
556
557         JScrollPane scrollPane = new JScrollPane();
558         scrollPane.setBounds(35, 223, 825, 213);
559         frmClientTp.getContentPane().add(scrollPane);
560
561         txtout = new JTextArea();
562         scrollPane.setViewportView(txtout);
563         txtout.setForeground(Color.GREEN);
564         txtout.setBackground(Color.GRAY);
565
566         JLabel lblEnterUserName = new JLabel("Enter User Name :");
567         lblEnterUserName.setHorizontalAlignment(SwingConstants.LEFT
                );
568         lblEnterUserName.setBounds(472, 33, 159, 29);
569         frmClientTp.getContentPane().add(lblEnterUserName);
570
571         input4 = new JTextField();
572         input4.setHorizontalAlignment(SwingConstants.CENTER);
573         input4.setFont(new Font("Dialog", Font.BOLD | Font.ITALIC,
                26));
574         input4.setColumns(10);
575         input4.setBounds(652, 33, 208, 38);
576         frmClientTp.getContentPane().add(input4);
577
578     }
579
580     static class EchoClient {
581
582         private DatagramSocket socket;
583         private InetAddress address;
584         private int port;
585
586         private byte[] buf = new byte[65535];
587
588         public EchoClient(InetAddress adr, int por) throws
                SocketException, UnknownHostException {
589             socket = new DatagramSocket();
590             address = adr;
591             port = por;
592         }
593
594         public String sendEcho(String msg) throws Exception {
595             buf = msg.getBytes();
596             DatagramPacket packet = new DatagramPacket(buf, buf.
                    length, address, port);
597             socket.send(packet);
598             buf = new byte[65535];
599             packet = new DatagramPacket(buf, buf.length);
600             socket.receive(packet);
601             String received = new String(packet.getData(), 0,
                    packet.getLength());
```

```
602                 return received;
603             }
604 
605             public void close() {
606                 socket.close();
607             }
608         }
609 
610 }
```